Inverting and Visualizing Features for Object Detection*

Carl Vondrick, Aditya Khosla, Tomasz Malisiewicz, Antonio Torralba
Massachusetts Institute of Technology
{vondrick,khosla,tomasz,torralba}@csail.mit.edu

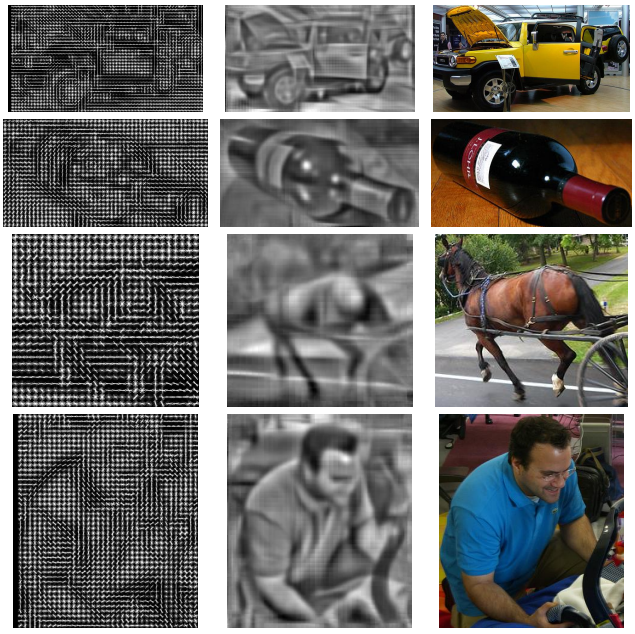
Abstract

This paper presents methods to visualize feature spaces commonly used in object detection. The tools in this paper allow a human to put on "feature space glasses" and see the visual world as a computer might see it. We found that these "glasses" allow us to gain insight into the behavior of computer vision systems. We show a variety of experiments with our visualizations, such as examining the linear separability of recognition in HOG space, generating high scoring "super objects" for an object detector, and diagnosing false positives. We pose the visualization problem as one of feature inversion, i.e. recovering the natural image that generated a feature descriptor. We describe four algorithms to tackle this task, with different trade-offs in speed, accuracy, and scalability. Our most successful algorithm uses ideas from sparse coding to learn a pair of dictionaries that enable regression between HOG features and natural images, and can invert features at interactive rates. We believe these visualizations are useful tools to add to an object detector researcher's toolbox, and code is available.

1. Introduction

A core building block for most modern recognition systems is a histogram of oriented gradients (HOG) [5]. While machines struggle to comprehend raw pixel values, HOG provides computers with a higher level representation of an image. The computational power of this representation has been substantially demonstrated by the community in object detection [3, 10, 19, 25, 32] as well as scene classification [22, 30] and motion tracking [2, 11].

Yet, the human vision system processes photons—not high dimensional vectors—making human interpretation of HOG features potentially counter-intuitive. As object detection researchers, we often spend considerable time staring at false positives and asking ourselves: why does our detector think there is a microwave flying in the sky?



(a) HOG (b) Inverse (This Paper) (c) Original

Figure 1: In this paper, we present several algorithms for inverting HOG descriptors back to images. The middle column is generated only from HOG features.

In this paper, we attempt to give humans a microscope into the world of HOG. We present four algorithms for visualizing and inverting HOG features back into natural images. Each algorithm has different trade-offs, varying in speed, accuracy, and scalability. Some of our algorithms use large databases; some are parametric. All of our algorithms are simple to use and understand.[1]

Our visualizations, shown in Fig.1, are intuitive for humans to grasp while still remaining true to the information stored inside each HOG feature, a claim we support with a user study. We found that this visualization power can give us insight into the behavior of object detectors. For example, when we invert the false positives for an object detector,

*This paper is a pre-print of our conference paper. Last modified December 23, 2012.

[1] Code is available at http://mit.edu/vondrick/ihog.

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
23 DEC 2012		**00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Inverting and Visualizing Features for Object Detection	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Massachusetts Institute of Technology,77 Massachusetts Avenue,Cambridge,MA,02139**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This paper presents methods to visualize feature spaces commonly used in object detection. The tools in this paper allow a human to put on ?feature space glasses? and see the visual world as a computer might see it. We found that these ?glasses? allow us to gain insight into the behavior of computer vision systems. We show a variety of experiments with our visualizations, such as examining the linear separability of recognition in HOG space, generating high scoring ?super objects? for an object detector, and diagnosing false positives. We pose the visualization problem as one of feature inversion, i.e. recovering the natural image that generated a feature descriptor. We describe four algorithms to tackle this task, with different trade-offs in speed accuracy, and scalability. Our most successful algorithm uses ideas from sparse coding to learn a pair of dictionaries that enable regression between HOG features and natural images, and can invert features at interactive rates. We believe these visualizations are useful tools to add to an object detector researcher?s toolbox, and code is available.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**	**Same as Report (SAR)**	**8**	

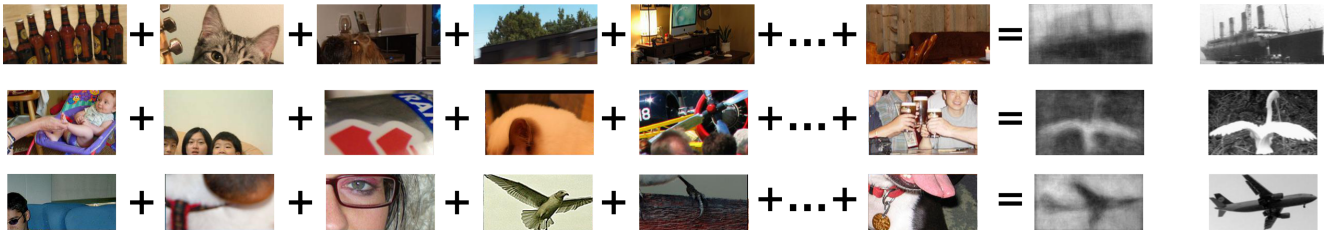


Figure 2: Inverting HOG features using exemplar LDA. We train an exemplar LDA model on the HOG descriptor we wish to invert and apply it to a large database. The left hand side of the above equation are the top detections, while the right hand side shows the average of the top 100. Even though all top detections are semantically meaningless, their average is close to the original image, shown on the right. Notice that all the top detections share structure with the original, e.g., the top left bottles create the smoke stack for the ship, and the middle right hands compose the wings for the bird.

we find the inversions look like true positives. This result suggests that the false positives are reasonable, and higher level reasoning may be necessary to solve object detection. By observing the visual world as object detectors see it, we can more clearly understand object recognition systems.

The contributions in this paper are two-fold. First, we offer four algorithms to invert HOG features. Second, we use our inversion algorithms to examine the behavior of object detectors. To this end, in section 2, we briefly review related work in reconstructing images given their feature descriptors. In section 3, we describe four algorithms for inverting and visualizing HOG features. Although we focus on HOG in this paper, our approach is general and can be applied to other features as well. In section 4, we evaluate the performance of our visualizations with a human study by asking subjects to identify objects given only their inverse. Finally, in section 5, we present a variety of experiments using HOG inversion to visualize the behavior of object detectors.

2. Related Work

There has been relatively little work in feature inversion so far. Torralba and Oliva, in early work [27], described a simple iterative procedure to recover images only given gist descriptors [21]. Weinzaepfel et al. [29] were the first to reconstruct an image given its keypoint SIFT descriptors [17]. Their approach obtains compelling reconstructions using a nearest neighbor based approach on a massive database. We encourage readers to see their full color reconstructions. However, their approach only focuses on sparse keypoint SIFT descriptors. Since most object detectors use a dense histogram of visual features, we instead present algorithms for inverting histogram features for detection, the most popular of which is HOG. Our algorithms are also quick, allowing for nearly real time visualization. d'Angelo et al. [6] further developed an algorithm to reconstruct images given only LBP features [4, 1]. Their method analytically solves for the inverse image and does not require a dataset. While [29, 6, 27] do a good job at reconstructing images from SIFT, LBP, and gist features, to our knowledge, this paper is the first to invert HOG.

This work also complements a recent line of papers that examine object detectors. Hoiem et al. [13] performed a large study analyzing the errors that object detectors make. Parikh and Zitnick [23] introduced a paradigm for human debugging of object detectors. Divvala et al. [7] analyze part-based detectors to determine the importance of each piece in the object detection stack. Tatu et al. [24] explored the set of images that generate identical HOG descriptors. Zhu et al. [33] try to determine whether we have reached Bayes risk for HOG. In this paper, we analyze object detectors by direct inspection: we visualize the world as computers see it by inverting HOG.

3. Feature Inversion Algorithms

Let $x \in \mathbb{R}^D$ be an image and $y = \phi(x)$ be the corresponding HOG feature descriptor. Since $\phi(\cdot)$ is a many-to-one function, no analytic inverse exists. Hence, we seek an image x that, when computed HOG on it, closely matches the original descriptor y:

$$\phi^{-1}(y) = \underset{x \in \mathbb{R}^D}{\operatorname{argmin}} \, ||\phi(x) - y||_2^2 \qquad (1)$$

Optimizing Eqn.1 is challenging. Although Eqn.1 is non-convex, we tried gradient-descent strategies by numerically evaluating the derivative in image space with Newton's method. Unfortunately, we observed poor results, likely because HOG is both highly sensitive to noise and Eqn.1 has frequent local minimas. In the remainder of this section, we present four different algorithms for inverting HOG.

3.1. Algorithm A: Exemplar LDA (ELDA)

Consider the top detections for the exemplar object detector [12, 19] for a few images shown in Fig.2. Although all top detections are false positives, notice that each detection captures some statistics about the query. Even though the detections are wrong, if we squint, we can see parts of the original object appear in each detection.

We use this simple observation to produce our first inversion algorithm. Suppose we wish to invert HOG feature y. We first train an exemplar LDA detector [12] for this query, $w = \Sigma^{-1}(y - \mu)$. We then score w against every sliding window on a large database. The HOG inverse is then simply the average of the top K detections in RGB space: $\phi_A^{-1}(y) = \frac{1}{K}\sum_{i=1}^{K} z_i$ where z_i is a top detection.

This method, although simple, produces surprisingly accurate reconstructions, even when the database does not contain the category of the HOG template. We note that this method may be subject to dataset bias [26] but could be overcome [15]. We also point out that a similar nearest neighbor method is used in brain research to visualize what a person might be seeing [20].

3.2. Algorithm B: Ridge Regression

Unfortunately, running an object detector across a large database is computationally expensive. In this section, we present a fast, parametric inversion algorithm.

Let $X \in \mathbb{R}^D$ be a random variable representing a gray scale image and $Y \in \mathbb{R}^d$ be a random variable of its corresponding HOG point. We define these random variables to be normally distributed on a $D + d$-variate Gaussian $P(X, Y) \sim \mathcal{N}(\mu, \Sigma)$ with parameters $\mu = [\mu_X \ \mu_Y]$ and $\Sigma = \begin{bmatrix} \Sigma_{XX} & \Sigma_{XY} \\ \Sigma_{XY}^T & \Sigma_{YY} \end{bmatrix}$. In order to invert a HOG feature y, we calculate the most likely image from the Gaussian P conditioned on $Y = y$:

$$\phi_B^{-1}(y) = \underset{x \in \mathbb{R}^D}{\operatorname{argmax}} P(X = x | Y = y) \tag{2}$$

It is well known that Gaussians have a closed form conditional mode:

$$\phi_B^{-1}(y) = \Sigma_{XY}\Sigma_{YY}^{-1}(y - \mu_Y) + \mu_X \tag{3}$$

Under this inversion algorithm, any HOG point can be inverted by a single matrix multiplication, allowing for inversion in under a second.

We estimate μ and Σ on a large database. In practice, Σ is not positive definite; we add a small uniform prior (i.e., $\hat{\Sigma} = \Sigma + \lambda I$) so Σ can be inverted. Since we wish to invert any HOG point, we assume that $P(X, Y)$ is stationary [12], allowing us to efficiently learn the covariance across massive datasets. We invert a arbitrary dimensional HOG point by marginalizing out unused dimensions.

3.3. Algorithm C: Direct Optimization

We found that ridge regression yields blurred inversions. Intuitively, since HOG is invariant to shifts up to its bin size, there are many images that map to the same HOG point. Ridge regression is reporting the statistically most likely image, which is the average over all shifts. This causes ridge regression to only recover the low frequencies of the original image.

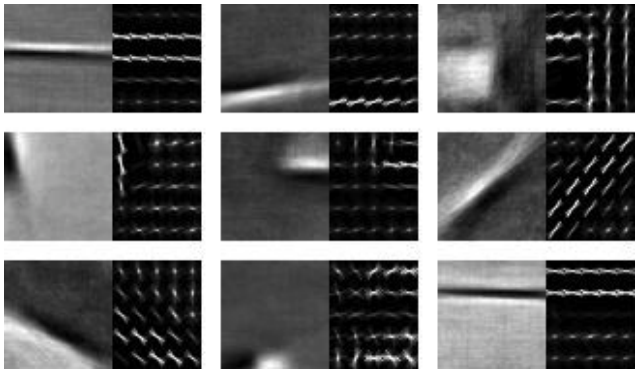


Figure 3: Some pairs of dictionaries for U and V. The left of every pair is the gray scale dictionary element and the right is the positive components elements in the HOG dictionary. Notice that the gray patches are correlated with the HOG patches.

We now provide an algorithm to recover the high frequencies. Let $U \in \mathbb{R}^{D \times K}$ be a natural image basis (e.g., the first K eigenvectors of $\Sigma_{XX} \in \mathbb{R}^{D \times D}$). Any image $x \in \mathbb{R}^D$ can be encoded by coefficients $\rho \in \mathbb{R}^K$ in this basis: $x = U\rho$. Since ridge regression only recovers the first few principal components of U, there is a residual term of high frequencies left to be recovered:

$$x = \sum_{i=1}^{K} U\rho_i = \text{Low} + \text{High} = \phi_B^{-1}(y) + \sum_{i=J}^{K} U\rho_i \tag{4}$$

where $\phi_B^{-1}(\cdot)$ was able to only recover J components. The goal of our third approach is to explicitly recover the high frequency components, i.e. the second term. We wish to minimize:

$$\phi_C^{-1}(y) = \underset{\rho \in \mathbb{R}^K}{\operatorname{argmin}} \left|\left|\phi(\lambda\phi_B^{-1}(y) + U\rho) - y\right|\right|_2^2 \tag{5}$$

for some hyperparameter $\lambda \in \mathbb{R}$. Empirically we found success optimizing Eqn.5 using coordinate descent on ρ with random restarts. We use an over-complete basis corresponding to sparse Gabor-like filters for U. We compute the eigenvectors of Σ_{XX} across different scales and translate smaller eigenvectors to form U.

3.4. Algorithm D: Paired Dictionary Learning

Direct optimization obtains highly accurate results, but since optimization requires computing HOG features on a large number of candidate images, convergence is slow. In our final algorithm, we propose a fast approximation.

Let $x \in \mathbb{R}^D$ be an image and $y \in \mathbb{R}^d$ be its HOG descriptor. The key observation is that if we write x and y in terms of bases $U \in \mathbb{R}^{D \times K}$ and $V \in \mathbb{R}^{d \times K}$ respectively, but with shared coefficients $\alpha \in \mathbb{R}^K$,

$$x = U\alpha \quad \text{and} \quad y = V\alpha \tag{6}$$

then inversion can be obtained by first projecting the HOG features y onto the HOG basis V, then projecting α into the natural image basis U:

$$\phi_D^{-1}(y) = U\hat{\alpha} \tag{7}$$

$$\text{where} \quad \hat{\alpha} = \underset{\alpha}{\text{argmin}} ||y - V\alpha|| \quad \text{s.t.} \quad ||\alpha||_1 \leq \lambda \tag{8}$$

Since efficient solvers for Eqn.8 exist [18, 16], we are able to invert HOG patches in under a second.

This paired dictionary trick requires finding appropriate bases U and V such that Eqn.6 holds. To do this, we solve a paired dictionary learning problem, inspired by recent superresolution sparse coding work [31, 28]:

$$\begin{aligned} &\underset{U,V,\alpha}{\text{argmin}} \sum_{i=1}^{N} \left(||x_i - U\alpha_i||_2^2 + ||\phi(x_i) - V\alpha_i||_2^2 \right) \\ &\text{s.t.} \quad ||\alpha_i||_1 \leq \lambda \; \forall i, \; ||U||_2^2 \leq \gamma_1, \; ||V||_2^2 \leq \gamma_2 \end{aligned} \tag{9}$$

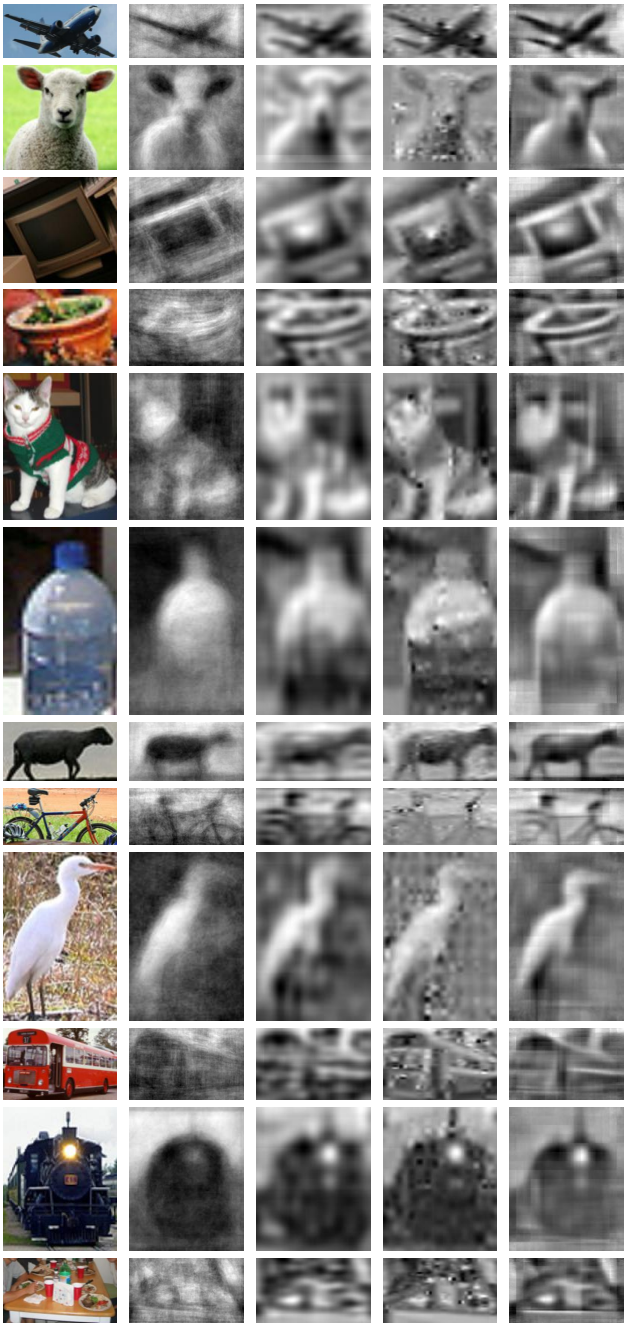
After a few algebraic manipulations, the above objective simplifies to a standard sparse coding and dictionary learning problem with concatenated dictionaries, which we optimize using SPAMS [18]. Optimization typically took a few hours on medium sized problems. We estimate U and V with a dictionary size $K \in O(10^3)$ and training samples $N \in O(10^6)$ from a large database. See Fig.3 for a visualization of the learned dictionary pairs.

Unfortunately, the paired dictionary learning formulation suffers on problems of nontrivial scale. In practice, we only learn dictionaries for 5×5 HOG templates. In order to invert a $w \times h$ HOG template y, we invert every 5×5 subpatch inside y and average overlapping patches in the final reconstruction. We found that this approximation works well in practice.

4. Evaluation

In this section, we evaluate our four inversion algorithms using both qualitative and quantitative measures. We use PASCAL VOC 2011 [8] as our dataset and we invert patches corresponding to objects. Any algorithm that required training could only access the training set. During evaluation, only images from the validation set are examined. The database for exemplar LDA excluded the category of the patch we were inverting to reduce the effect of biases.

We show our inversions in Fig.4 for a few object categories. Exemplar LDA and ridge regression tend to produce blurred visualizations. Direct optimization recovers high frequency details at the expense of extra noise. Paired dictionary learning produces the best visualization for HOG descriptors. By learning a sparse dictionary over the visual world and the correlation between HOG and natural images, paired dictionary learning recovered high frequencies without introducing significant noise.



(a) Original (b) ELDA (c) Ridge (d) Direct (e) PairDict

Figure 4: We show the results for all four of our inversion algorithms on held out image patches on similar dimensions common for object detection. In general, exemplar LDA produces grainy inversions. Ridge regression is blurry, but fast. Direct optimization is able to recover high frequencies at the expense of extra noise; notice the eyes on the sheep and cat, and details on the bus. Paired dictionary learning often perceptually performs the best, striking a middle ground between crisp and blurry.

Category	ELDA	Ridge	Direct	PairDict
aeroplane	**0.634**	**0.633**	0.596	0.609
bicycle	0.452	**0.577**	0.513	0.561
bus	0.627	**0.632**	0.587	0.585
cat	**0.749**	0.712	0.687	0.705
cow	**0.720**	0.663	0.632	0.650
horse	**0.686**	0.633	0.586	0.635
tvmonitor	**0.711**	0.640	0.638	0.629
Mean	**0.671**	0.656	0.620	0.637

Table 1: We evaluate the performance of our inversion algorithm by comparing the inverse to the ground truth image using the mean normalized cross correlation. Higher is better; a score of 1 is perfect. In general, exemplar LDA does slightly better at reconstructing the original pixels.

SIFT Comparison: We compare our HOG inversions against SIFT reconstructions on the INRIA Holidays dataset [14]. Fig.5 shows a qualitative comparison between paired dictionary learning and Weinzaepfel et al. [29]. Notice that HOG inversion is more blurred than key point SIFT since HOG is histogram based.

Dimensionality: HOG inversions are sensitive to the dimensionality of their templates. For medium (10×10) to large templates (40×40), we obtain reasonable performance. But, for small templates (5×5) the inversion is blurred. Fig.6 shows examples as the HOG descriptor dimensionality changes.

In the remainder of this section, we evaluate our algorithms under two benchmarks: first, an inversion metric that measures how well our inversions reconstruct the original images, and second, a visualization challenge conducted on Amazon Mechanical Turk designed to determine how well people can infer the original category from the inverse. The first experiment measures the algorithm's reconstruction error, while the second experiment analyzes the recovery of high level semantics.

4.1. Inversion Benchmark

We consider the inversion performance of our algorithm: given a HOG feature y, how well does our inverse $\phi^{-1}(y)$ reconstruct the original pixels x for each algorithm? Since HOG is invariant up to a constant shift and scale, we score each inversion against the original image with normalized cross correlation. Our results are shown in Tab.1. Overall, exemplar LDA does the best at pixel level reconstruction.

4.2. Visualization Benchmark

While the inversion benchmark evaluates how well the inversions reconstruct the original image, it does not capture the high level content of the inverse: is the inverse of a sheep still a sheep? To evaluate this, we conducted a study on Amazon Mechanical Turk. We sampled 2,000 windows

(a) HOG (Us) (b) Original (c) Keypoint SIFT [29]

Figure 5: We compare our paired dictionary learning approach on HOG with the algorithm of [29] on SIFT. Since HOG is invariant to color, we are only able to recover a grayscale image. Furthermore, our blurred inversion shows that HOG is a more coarse descriptor than keypoint SIFT.

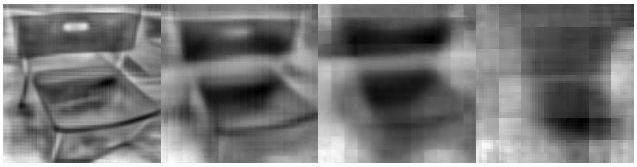


Figure 6: Our inversion algorithms are sensitive to the HOG template size. Larger templates are easier to invert since they are less invariant. We show how performance degrades as the template becomes smaller. Dimensions in HOG space shown: 40×40, 20×20, 10×10, and 5×5.

corresponding to objects in PASCAL VOC 2011. We then showed participants an inversion from one of our algorithms and asked users to classify it into one of the 20 categories. Each window was shown to three different users. Users were required to pass a training course and qualification exam before participating in order to guarantee users unserstood the task. Users could optionally select that they were not confident in their answer. We also compared our algorithms against the standard black-and-white HOG glyph popularized by [5].

Our results in Tab.2 show that paired dictionary learning and direct optimization provide the best visualization of HOG descriptors for humans. Ridge regression and exemplar LDA performs better than the glyph, but they suffer from blurred inversions. Human performance on the HOG glyph is generally poor, and participants were even the slowest at completing that study. Interestingly, the glyph does the best job at visualizing bicycles, likely due to their unique circular gradients. Overall, our results suggest that visualizing HOG with the glyph is misleading, and using richer diagrams is useful for interpreting HOG vectors.

There is strong correlation with the accuracy of humans classifying the HOG inversions with the performance of HOG based object detectors. We found human classification accuracy on inversions and the state-of-the-art object detection AP scores from [9] are correlated with a Spearman's rank correlation coefficient of 0.77. This result sug-

Category	ELDA	Ridge	Direct	PairDict	Glyph	Expert
bicycle	0.327	0.127	0.362	0.307	**0.405**	0.438
bird	0.364	0.263	**0.378**	0.372	0.193	0.059
bottle	0.269	0.282	0.283	**0.446**	0.312	0.222
car	0.397	0.457	**0.617**	0.585	0.359	0.389
cat	0.219	0.178	**0.381**	0.199	0.139	0.286
chair	0.099	0.239	0.223	**0.386**	0.119	0.167
table	0.152	0.064	0.162	**0.237**	0.071	0.125
horse	0.260	0.290	0.354	**0.446**	0.144	0.150
motorbike	0.221	0.232	**0.396**	0.224	0.298	0.350
person	0.458	0.546	0.502	**0.676**	0.301	0.375
sofa	0.138	0.100	0.162	**0.293**	0.104	0.000
Mean	0.282	0.258	0.355	**0.383**	0.191	0.233

Table 2: We evaluate visualization performance across twenty PASCAL VOC categories by asking Mechanical Turk workers to classify our inversions. Numbers are percent classified correctly; higher is better. Chance is 0.05. Glyph refers to the standard black-and-white HOG diagram popularized by [5]. Paired dictionary learning provides the best visualizations for humans. Expert refers to PhD students in computer vision performing the same visualization challenge with HOG glyphs. Notice that even HOG experts can benefit from paired dictionary learning. Interestingly, the glyph is best for bicycles.

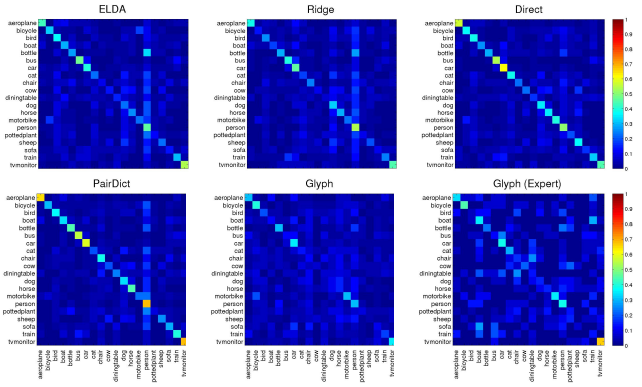


Figure 7: We show the confusion matrices for each of our four algorithms as well as the standard HOG black-and-white glyph visualization. The vertical axis is the ground truth category and the horizontal axis is the predicted category. Notice that common confusions are similar to errors caused made by detectors. The expert confusion matrix refers to the workers who are computer vision PhD students.

gests that humans can predict the performance of object detectors by only looking at HOG visualizations.

Fig.7 shows the classification confusion matrix for all algorithms. Participants tended to make the same mistakes that object detectors make. Notice that bottles are often confused with people, motorbikes with bicycles, and animals with other animals. Users incorrectly showed a strong prior that the inversions were for people, evidenced by a bright vertical bar in the confusion matrix.

(a) Human Vision (b) HOG Vision

Figure 8: HOG inversion reveals the world that object detectors see. The left shows a man standing in a dark room. If we compute HOG on this image and invert it, the previously dark scene behind the man emerges. Notice the wall structure, the lamp post, and the chair in the bottom right hand corner.

We also asked computer vision PhD students to classify HOG glyphs in order to compare Mechanical Turk workers with experts in HOG. Our results are summarized in the last column of Tab.2. HOG experts performed slightly better than common people on the glyph challenge, but experts on glyphs did not beat common people on other visualizations. This result suggests that our algorithms produce more intuitive visualizations even for object detection researchers.

5. Experiments

The underlying motivation of this paper has been to develop feature inversion algorithms and use these visualizations to analyze object detectors. In this section, we present several experiments that use our inversions to put on "HOG glasses" to analyze the behvaior of HOG.

How Detectors See the World: In our first experiment, we attempt to reveal how object detectors see the visual world. Fig.8a shows a normal photograph of a man, but Fig.8b shows how HOG sees the same man. Since HOG is invariant to illumination changes, the background of the scene, invisible to the human eye, materializes, demonstrating the clutter that HOG catches.

Top False Positives: Seeing the world through the eyes of HOG can be helpful for understanding object detector errors. We train a single mixture component using SVM and HOG. Fig.9 shows the top false detections for a few categories and their inverses. Notice that the inversions look like the positive class while the original image patch does not. This experiment suggests that the false positives that

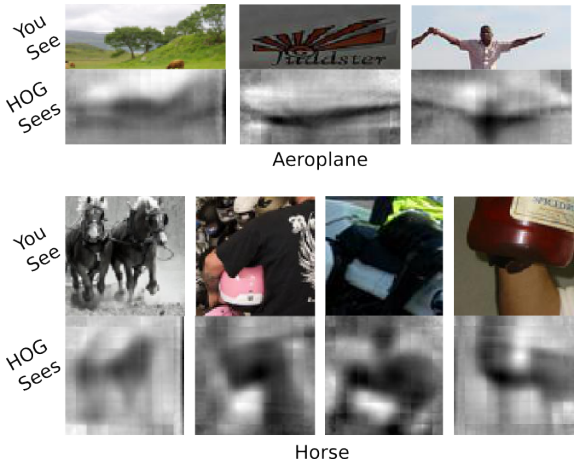


Figure 9: We trained a single mixture component model with SVM for a few classes. This figure shows some of the top false positives and their inversions. Notice that the inversions look like true positives—the airplane's wings and body, and horse's legs and torso appear in the inversion, but not necessarily in the original image.

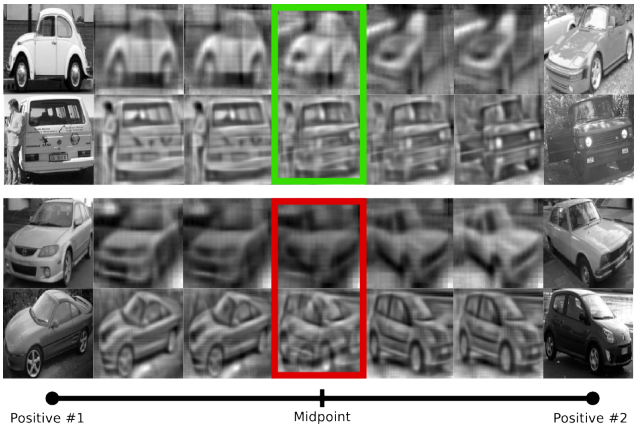


Figure 10: We linearly interpolate between examples in HOG space and invert its path. First two rows: occasionally, the interpolation of two examples is still in the positive class even under extreme viewpoint change. Last two rows: frequently, however, the midpoint is no longer the positive. This demonstrates that a single linear separator in HOG space is insufficient for perfect object detection.

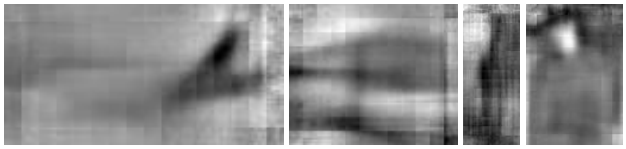


Figure 11: We invert the positive components of a few root templates from the deformable parts model [10]. Notice the airplane tail wing, the right facing bus, the typical bottle, and a person leaning his head.

object detectors predict in HOG space are reasonable and higher level reasoning may be necessary to improve object recognition performance.

Interpolation in Feature Space: Since object detection is computationally expensive, most state-of-the-art object detectors today depend on linear classifiers. Fig.10 analyzes whether recognition is linear separable in HOG space by inverting the midpoint between two positive examples. Not surprisingly, our results show that frequently the midpoint no longer resembles the positive class. Since linear classifiers assume that the midpoint of any positive example is also a positive, this result indicates that perfect car detection is not possible with a single linear separator in HOG space. Car detection may be solvable with view based mixture components, motivating much recent work in increasing model complexity [19, 10].

Prototypical Objects: We analyze an object detector's prototypical example of an object. Fig.11 shows the positive component of the weight vector for a few object detectors trained with [10]. The prototypes highlights the parts of objects that each detector finds discriminative. Notice how that prototypes look similar to the average of the class.

Super Objects: In Fig.12, we examine how the appearance of objects change as we make an object "more positive" or "more negative." We move perpendicularly to the class decision boundary in HOG space. As the object becomes more and more positive, the key gradients become more pronounced, but if the object is downgraded towards the negative world, the object starts looking like noise. This experiment gives an intuitive visualization of what each object detector finds important.

6. Conclusion

We have presented four algorithms for inverting and visualizing features for object detection. While this paper has focused on HOG, our algorithms are general and can be applied to any feature descriptor. We evaluated our method against a difficult dataset with a large human study and we presented several experiments that use feature inversion in order to see the world through the eyes of an object detector. Our best performing algorithm, paired dictionary learning, uses ideas from sparse coding to regress between feature descriptors and their natural images. Since efficient solvers for sparse coding now exist, we are able to invert features at nearly interactive rates. We hope that others find these visualizations useful in their own research.

Acknowledgements: We thank Hamed Pirsiavash, Joseph Lim, and the MIT CSAIL vision group. Funding for this research was provided by a NSF GRFP to CV and a

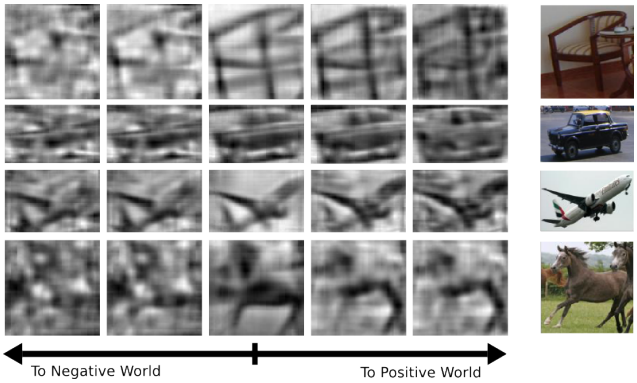


Figure 12: We train single component, linear SVM object detectors with HOG for a variety of categories and translate in HOG space orthogonal to the decision hyperplane. Moving towards the right is making the object more positive and to the left is making it more negative. The full color image on the right is the original image. Moving towards the positive world causes the discriminative gradients of the example to increase, and moving to the negative world causes the example to become more like background noise.

Google research award, ONR MURI N000141010933 and NSF Career Award No. 0747120 to AT.

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. Freak: Fast retina keypoint. In *CVPR*, 2012. 2

[2] S. Avidan. Ensemble tracking. *PAMI*, 2007. 1

[3] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009. 1

[4] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. Brief: Binary robust independent elementary features. *ECCV*, 2010. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 1, 5, 6

[6] E. d'Angelo, A. Alahi, and P. Vandergheynst. Beyond bits: Reconstructing images from local binary descriptors. *ICPR*, 2012. 2

[7] S. Divvala, A. Efros, and M. Hebert. How important are deformable parts in the deformable parts model? *Technical Report*, 2012. 2

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 4

[9] P. Felzenszwalb, R. Girshick, and D. McAllester. Cascade object detection with deformable part models. In *Computer vision and pattern recognition (CVPR), 2010 IEEE conference on*, pages 2241–2248. IEEE, 2010. 5

[10] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2010. 1, 7

[11] H. Grabner, M. Grabner, and H. Bischof. Real-time tracking via on-line boosting. In *BMVC*, 2006. 1

[12] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. *ECCV*, 2012. 2, 3

[13] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. *ECCV*, 2012. 2

[14] H. Jegou, M. Douze, and C. Schmid. Hamming embedding and weak geometric consistency for large scale image search. *Computer Vision–ECCV 2008*, pages 304–317, 2008. 5

[15] A. Khosla, T. Zhou, T. Malisiewicz, A. Efros, and A. Torralba. Undoing the damage of dataset bias. 2012. 3

[16] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. *NIPS*, 2007. 4

[17] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, 1999. 2

[18] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *ICML*, 2009. 4

[19] T. Malisiewicz, A. Gupta, and A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011. 1, 2, 7

[20] S. Nishimoto, A. Vu, T. Naselaris, Y. Benjamini, B. Yu, and J. Gallant. Reconstructing visual experiences from brain activity evoked by natural movies. *Current Biology*, 2011. 3

[21] A. Oliva, A. Torralba, et al. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 2006. 2

[22] M. Pandey and S. Lazebnik. Scene recognition and weakly supervised object localization with deformable part-based models. In *ICCV*, 2011. 1

[23] D. Parikh and C. Zitnick. Human-debugging of machines. In *Workshop on Computational Social Science and the Wisdom of Crowds, NIPS*, 2011. 2

[24] A. Tatu, F. Lauze, M. Nielsen, and B. Kimia. Exploring the representation capabilities of the hog descriptor. In *ICCV Workshops*, 2011. 2

[25] P. Torr and A. Zisserman. Latent svms for human detection with a locally affine deformation field. 1

[26] A. Torralba and A. Efros. Unbiased look at dataset bias. In *CVPR*, 2011. 3

[27] A. Torralba and A. Oliva. Depth estimation from image structure. *PAMI*, 2002. 2

[28] S. Wang, L. Zhang, Y. Liang, and Q. Pan. Semi-coupled dictionary learning with applications to image super-resolution and photo-sketch synthesis. In *CVPR*, 2012. 4

[29] P. Weinzaepfel, H. Jégou, and P. Pérez. Reconstructing an image from its local descriptors. In *CVPR*, 2011. 2, 5

[30] J. Xiao, J. Hays, K. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 1

[31] J. Yang, J. Wright, T. Huang, and Y. Ma. Image super-resolution via sparse representation. *Transactions on Image Processing*, 2010. 4

[32] Q. Zhu, M. Yeh, K. Cheng, and S. Avidan. Fast human detection using a cascade of histograms of oriented gradients. In *CVPR*, 2006. 1

[33] X. Zhu, C. Vondrick, D. Ramanan, and C. Fowlkes. Do we need more training data or better models for object detection? *BMVC*, 2012. 2